AFRL-RZ-WP-TP-2008-2183

MULTI-CORE PROCESSORS: AN ENABLING TECHNOLOGY FOR EMBEDDED DISTRIBUTED MODEL-BASED CONTROL (POSTPRINT)

Alireza Behbahani, Nathan Gibson, Murali Rangarajan, and Dewey Benson

Structures and Controls Branch
Turbine Engine Division

JULY 2008

AIR FORCE RESEARCH LABORATORY
PROPULSION DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7251
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
July 2008	Conference Paper Postprint	02 January 2008 – 21 July 2008

4. TITLE AND SUBTITLE
MULTI-CORE PROCESSORS: AN ENABLING TECHNOLOGY FOR EMBEDDED DISTRIBUTED MODEL-BASED CONTROL (POSTPRINT)

5a. CONTRACT NUMBER
In-house

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62203F

6. AUTHOR(S)
Alireza Behbahani (AFRL/RZTS)
Nathan Gibson (GE Aviation)
Murali Rangarajan (Honeywell Aerospace, MN)
Dewey Benson (Honeywell Aerospace, AZ)

5d. PROJECT NUMBER
3066

5e. TASK NUMBER
03

5f. WORK UNIT NUMBER
306603TM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Structures and Controls Branch (AFRL/RZTS)
Turbine Engine Division
Air Force Research Laboratory, Propulsion Directorate
Wright-Patterson Air Force Base, OH 45433-7251
Air Force Materiel Command, United States Air Force

GE Aviation
Cincinnati, OH 45215

Honeywell Aerospace
Golden Valley, MN 55422

Honeywell Aerospace
Tempe, AZ 85284

8. PERFORMING ORGANIZATION REPORT NUMBER
AFRL-RZ-WP-TP-2008-2183

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Propulsion Directorate
Wright-Patterson Air Force Base, OH 45433-7251
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RZTS

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RZ-WP-TP-2008-2183

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
Conference paper presented at the 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, 21 –23 July 2008, in Hartford, CT. The PowerPoint slideshow presented at the conference is an attachment within the digital Adobe Acrobat .pdf report file.

PAO Case Number: WPAFB 08-4629; Clearance Date: 17 Jul 2008. The U.S. Government is joint author of this work and has the right to use, modify, reproduce, release, perform, display, or disclose the work.

14. ABSTRACT
Multi-core processing technology has been developing rapidly and multi-core solutions are becoming increasingly prevalent in industry. The potential impact and disruption of multi-core technologies to centralized computationally intensive applications, such as engine control, may be significant. With the ability to efficiently partition code between separate processing elements, whilst sharing a common I/O space, a new generation of process partitioning, a thread pipelining becomes possible. In this paper we briefly summarize the requirements and trends for FADEC based control applications and then discuss the emerging capabilities and challenges of multi-core processing technology in the context of the developing FADEC environment, presenting a hypothetical realization of an example application. Finally we discuss the application of Time-Triggered architectural techniques to the multi-core problem. The intersection of these technology areas may present some promising architectural options for disturbed based control applications of the future.

15. SUBJECT TERMS
Gas turbine, FADEC, Multi-core processing technology, disturbed based control applications, Multicores, CPU, Parallel Compilers, Asymmetric Multiprocessing, Inter-Process Communication, Thread-level parallelism

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Alireza R. Behbahani
Unclassified	Unclassified	Unclassified	SAR	18	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Multi-Core Processors: An Enabling Technology for Embedded Distributed Model-Based Control

Alireza Behbahani[1]
Air Force Research Labaratory (AFRL), Wright-Patterson AFB, OH, 45433, USA

Nathan Gibson[2]
GE Aviation, Cincinnati, OH 45215-198, USA

Murali Rangarajan[3]
Honeywell Aerospace, Golden Valley, MN, 55422, USA

and

Dewey Benson[4]
Honeywell Aerospace, Tempe, AZ, 85284, USA

Multi-core processing technology has been developing rapidly and multi-core solutions are becoming increasingly prevalent in industry. The potential impact and disruption of multi-core technologies to centralized computationally intensive applications, such as engine control, may be significant. With the ability to efficiently partition code between separate processing elements, whilst sharing a common I/O space, a new generation of process partitioning, a thread pipelining becomes possible. In this paper we briefly summarize the requirements and trends for FADEC based control applications and then discuss the emerging capabilities and challenges of multi-core processing technology in the context of the developing FADEC environment, presenting a hypothetical realization of an example application. Finally we discuss the application of Time-Triggered architectural techniques to the multi-core problem. The intersection of these technology areas may present some promising architectural options for disturbed based control applications of the future.

I. Introduction

The future innovative engine systems, while increasing performance, often necessitate better engine reliability and operability, thus requiring control system developers to deliver robustness that meets rising standards. Engine operability has many requirements including controls, performance, HPC, and fan for engine design. Additionally, these challenges must meet demanding schedules with fewer system prototypes at lower costs while maintaining high quality systems and software.

Aircraft engines and control systems represent a complex mechatronics system with non-linear interdependencies among different physical domains, mechanics and electronics. The operation characteristics of aircraft engines can change with engine-to-engine quality variation and aging of components over time, depending on the engine operating conditions and service history. Because of the complex flow fields and component interaction in a modern gas turbine engine, these engines require extensive testing to validate performance, control, and stability.

[1] Senior Aerospace Engineer, Propulsion Directorate, 1950 Fifth St., Bldg 18D-133.
[2] Military Controls Program Leader, One Neumann Way MD BBC-5.
[3] Senior Research Scientist, Honeywell Advanced Technology Platform Systems.
[4] Technology Fellow, Honeywell Advanced Technology Platform Systems.

Developing complex adaptive control systems and software for embedded real-time Model-Based Controls and Engine Health Management (EHM) is a daunting challenge for control designers and system engineers of advanced turbine engines. Such systems require more processing power[1,2] and close integration with embedded control systems. As the complexity of software increases, control architectures should rely on principles and technologies that help to reduce burden on development teams, simplify design and integration and reduce engine operating costs. The objective of this paper is to analyze how multi-core processors can contribute to integration of new capabilities into a FADEC.

II. Multicores and Aerospace Control Aplications

A. Multi-core and many-core processors

Multi-core processing refers to two or more processors on a single piece of silicon. Multi-core processors usually share a common interface to the external world, and may have either shared or independent cache memory for each processor, or some of both (Figure 1).

Multicore CPU designs are a result of complexity and limits introduced by instruction-level parallelism and high energy consumption due to clocking frequency scaling. Multi-threading is a logical solution as different applications can be executed on different processors efficiently, which is efficient with loosely coupled and non-real-time applications. For many-cores that have tens or hundreds of cores the novel methods for SW design, problem parallelization and operating systems will be essential for efficient use of available computing resources. Processing efficiency depends also on the type of the application, because not all computing problems can be parallelized.

Efficient sharing of cache and off-chip bandwidth, scalability, multithread synchronization and simplification of parallel programming are keys to widespread use of multi-cores in embedded computing applications. Deep understanding automated software design tools and above mentioned issues are required for productive design and certification of complex aerospace control systems.

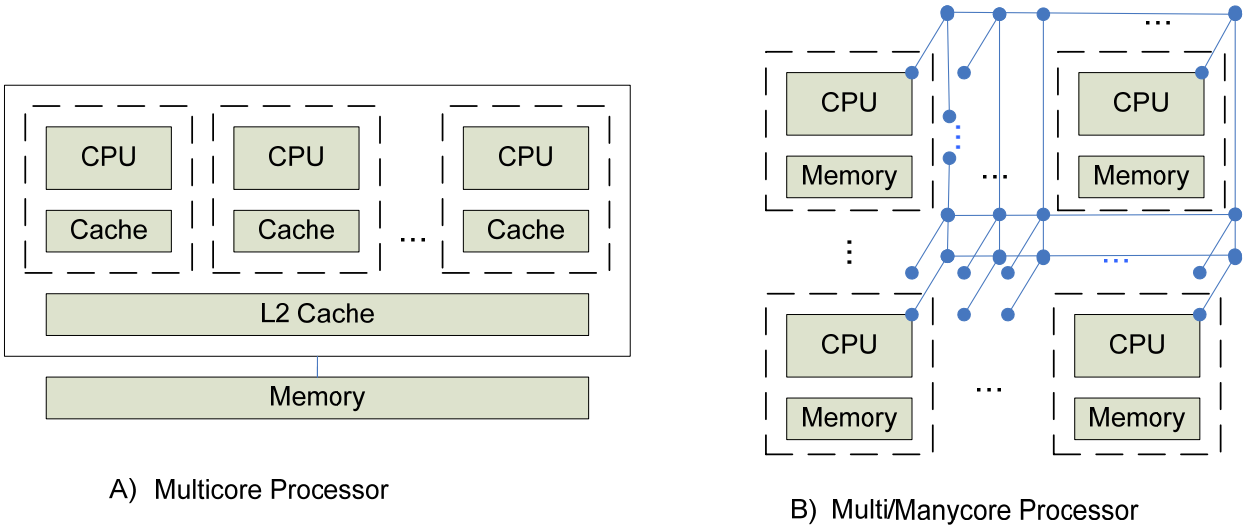

Figure 1. Multi core processing (image adapted from WindRiver's whitepaper[3])

B. FADEC design and multicore processors

Multi-cores can reduce hardware footprint, power dissipation and weight in applications where I/O processing, displays, low-level control loops and higher level functions such as Model-Based Control (MBC) and Engine Health Management (EHM) must be integrated on-board. All of these benefits lead to the impression that higher computational density can be achieved by delivering more computing power in a smaller package and addressing heat dissipation using conventional air or conduction cooling.

The use of on-board model based controls and real-time predictive health and fault monitoring is likely to drive the desire to use multi-core processing for FADEC applications. Without multi-cores, increase in processing requirements could result in growth of the size and weight of future FADECs.

In aerospace applications, a multi-core approach can also support additional redundancy and robustness in critical applications, especially if they are integrated into a distributed embedded computing system. By using a networked system with partitioning of the network media (e.g. based on time-triggered communications or high-speed TTEtherent[4]), functions can reside anywhere in the network and run synchronously on remote cores. Beyond the centralized FADEC with a multi-core processor, multiple processors in a distributed control system networked with time-triggered communications can reduce the programming effort, application code complexity, and verification/validation/qualification efforts for a given application.

III. Engine Control System Requirements

The engine controls function is required to perform engine starting, set engine thrust in response to command, prevent limit exceedance, and power manage under faulted conditions. This is accomplished via a Full Authority Digital Engine Control (FADEC), a computer that implements control laws/schedules via the engine control hardware such as sensors and actuators. Engine thrust must be maintained at both steady state and transient conditions, and the engine control must ensure that the turbomachinery stays within operability limits (Fig. 1). Additionally, the control strategy to achieve the thrust setting must take into account power offtakes, air bleed for other engine or aircraft needs, clearances, and thermal management. Of primary importance is fuel flow and variable geometry during transient operation (accels, decels, augmentor light-off, etc), while still maintaining limit protection (stall margins, blowout, over-speed, over-temps, etc) as indicated in Fig. 2.

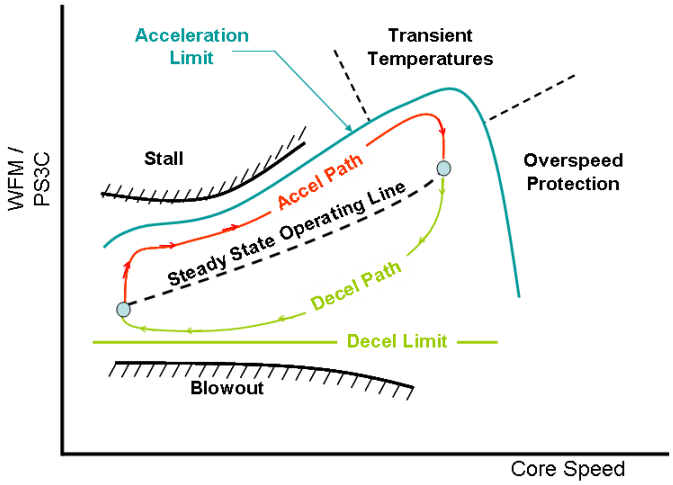

Figure 1. Transient Operability

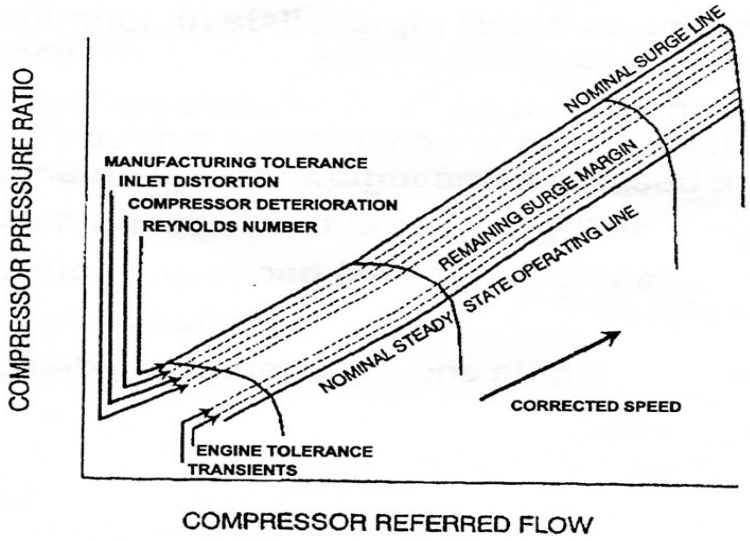

Figure. 2: Stall Margin Reduction

A. Advanced Control Strategies

As on-engine computing capability increases, more advanced control strategies can be utilized in the demanding FADEC hardware and software environment, enabling better controls performance and capacity. The aerospace community has investigated such techniques for years[5]. Embedded software and electronics are used across other industries' functional areas, replacing or simplifying mechanical and hydraulic actuation systems of controls. Control system designers that use embedded control system FADEC software face difficult challenges. In addition to the need to reduce cost, minimize obsolescence and tighten schedules, embedded FADEC control system software designers must provide predictable performance and real-time execution. This only makes the integration of advanced control techniques more challenging, however the potential benefits of advanced controls are significant.

B. Embedded Models

A key enabler of advanced control techniques is the embedded engine model (EEM). An aerothermodynamic EEM could take one of many different forms, such as a component level model (CLM), a linear engine model (LEM), a "partials" [derivative] model, full cycle model, etc, which have varying degrees of fidelity and capability. In a model-based control implementation, an EEM would likely be "tracked", meaning tuned in real-time to match actual engine conditions. FADEC implementation of an EEM faces challenges such as real-time execution and load balancing with other necessary control software, but model usage in the FADEC can be extensive, for model-based control and diagnostics.

C. Model Predictive Control (MPC)

In general, MPC is aimed at using online model-based optimization to improve some performance metric of a system close to entitlement, subject to critical safety, operability and hardware constraints. MPC is multivariable, and it employs dynamic constrained optimization under both steady-state and transient operation, thus allowing the control to adapt to specific engines accounting for variation, deterioration, and faults, exploit multivariable interaction, and optimize performance for various optimization modes. As such, MPC provides the most benefits when there are (1) multiple actuators/inputs to manipulate, (2) multiple constraints that need to be traded off against the optimization objectives, and/or (3) slow dynamics for which anticipation of oncoming constraints is critical. With these considerations, MPC is quite beneficial for system-level controls such as overall engine-level controls, flight-level controls, plant supervisory controls, etc. However for lower-level controls with higher bandwidth requirements, MPC becomes more difficult to use because (1) high bandwidth drives real-time implementation issues, and (2) the use of long prediction horizons (nominally the strength of MPC) can become detrimental to achieving good high-frequency response.

D. Multivariable Control

As the requirements placed upon modern turbine engines increase in complexity, multivariable control techniques become more attractive to the control designer. Multi-input-multi output (MIMO) approaches consider the interactions and "off-diagonal" effects between control loops, and can have a significant positive effect on engine control Critical To Quality (CTQs). However, as the size of the MIMO controller increases, the computing requirements tend to increase exponentially. As multivariable control theory and its application(s) to turbine engine control is well established, FADEC computing capability is the key limiting factor to full-scale implementation.

E. Active Control

Active control of clearances, stall, vibrations, acoustics, and emissions from turbine engines promises substantial benefits over current designs in terms of performance, operability, life, and environmental impact. While active control concepts have existed for many years, successful transition into production systems has been hindered by a lack of key enablers, such as high-bandwidth sensing and actuation systems and high-throughput processors in digital engine control systems. As an example, active stall/surge control would utilize high-bandwidth pressure/airflow sensors for feedback control of high bandwidth variable geometry actuators for stabilizing control resulting in quicker recovery from a stall or impending stall or ability to intentionally continuously operate in a stall region. Opportunities exist for implementation of such technology given adequate FADEC processing capability for these high-bandwidth closed-loop control algorithms.

F. Diagnostics / PHM

Real-time diagnostics utilizing a tracked EEM has been a focus of study for some time in the aviation gas turbine community. There are numerous approaches to this problem, including Kalman, state estimation, neural-fuzzy, etc. Damage detection, deterioration estimation, parts life tracking, and component health are just some of the applications of current PHM approaches. In an on-wing FADEC application, these approaches would generally rely upon accurate, real-time engine modeling with tracking capability. PHM solutions could improve and/or exhibit greater system-level confidence as model runtime capability increases. The design space for PHM technology would grow substantially as FADEC computing capability increases, providing numerous benefits to safety, fleet readiness, and maintenance costs.

G. Distributed Control

A topic of recent heavy interest in the engine control community is distributed control; a non-centralized approach that uses advanced communication, processing, and architecture to provide system benefits. The driving force for distributed systems for engine controls are the potential to reduce the size, weight and cost of wiring harnesses, simplification of system upgrades, distribution of computational burden, increased robustness against faults/damage, reduced cost of ownership, improved fault diagnostics, and obsolescence mitigation. Elements of distributed engine control technologies have been in development since the early 1990's, and managing the paradigm shift from centralized engine control to an architecture based on a distributed architecture utilizing open system standards is an industry-wide effort[12]. Many of the foundational technologies are under development in different sectors of the industry including high-temperature electronics, lightweight components, and advanced communication protocols. Computing horsepower will only increase the capability of distributed systems; multi-core processing has the potential to be a key enabler to meeting distributed control baseline requirements, such as redundancy, configurability, effector proximity, and component size.

IV. Taking Advantage of Dual and Multi-core Technology in FADEC

A. Trends in multicore processors

Multicore processors were developed by the processor vendors as a solution to the problems (such as increased heat dissipation) resulting from increased clock rates. This approach allows more work to be accomplished by spreading the work among more cores rather than by the same core working faster. The multicore approach was enabled by improvements to the manufacturing process technology, with more numbers of cores likely in the future.

Most multicore processors have homogeneous cores, a notable exception being IBM's Cell™ processor[8]. Processors with heterogeneous cores are designed for specific markets such as gaming. Over the last couple of years, many dual- and quad-core processors have been released. The trend is towards larger numbers of cores on a single processor. For example, Freescale's QorIQ™ product line, which is based on the 45nm technology, has just been announced, with the 8-core P4080 to be released by mid-2009. The QorIQ architecture is said to be scalable to 32 cores. Tilera Corporation is currently shipping a 64-core tile-architecture processor called TILE64™. The architecture can presumably scale to hundreds or even thousands of cores.

Another trend in the multicore processors is the integration of memory and I/O controllers on the processor itself, eliminating the traditional north and south bridges. Typical I/O controllers include GbE, PCI-e and Rapid I/O, with 18 to 24 SERDES channels typically available for their use. Both the QorIQ and TILE64 processors follow this approach.

These high levels of component integration on a single chip presents some unique communication challenges. Where as a simple bus or cross-bar was sufficient for on-chip communications, such an approach does not scale with increased numbers of cores. Therefore, newer architectures employ sophisticated mechanisms to connect the various components – The CoreNet™ fabric on the QorIQ, and Tilera's iMESH™ on the TILE64.

In spite of the increased numbers of cores, most multicore processors have at most two memory controllers due to constraints on the numbers of pins. This implies that the memory bandwidth is not keeping pace with the compute power. While this is not an issue (yet) for most systems, this is fast becoming a source of concern for high-criticality systems such as FADECs that need guaranteed memory bandwidths for their threads. The choice of processors that provide features enhancing data path separation is critical. Features of import include partitionable caches; ability to couple memory controllers with cores or threads; partitionable, prioritized internal communications mechanisms; and deterministic communication times along the various internal data paths.

B. Multicore technology and FADEC: Space, Weigth and Power (SWaP) constraints

Using a dual-core or multi-core processor has many advantages, including boosting the system's multitasking computing capability. Since each core has its own cache, each with an independent interface to the front side bus, and the operating system (OS) has sufficient resources to handle intensive tasks in parallel, which provides a noticeable processing improvement.

Optimization for the dual / multi core processor requires all the operating system and applications running on the computer to support a technology called thread-level parallelism (TLP). Thread-level parallelism is the part of the OS or application that runs multiple threads simultaneously, where threads refer to the part of a program that can execute independently of other parts. The shift to dual or multi core applications, and impact on performance for model-based control and other integrated high level FADEC functions will depend on the capability to parallelize tasks and take advantage of available processing power.

More and more processing power is required to do the control and reasoning tasks, which require higher processing capability for the processors. In addition, the faster clock of the single-core device requires faster memory, which further increases the power consumption and requires special packaging to dissipate the heat. Due to soaring power demands and heating problems associated with the perpetual pursuit of speed, microprocessor vendors are forced to re-evaluate their processor strategies and look at other ways of gauging performance. For military applications, the FADEC is cooled by fuel, and when operating in hot locations, the aircraft may have to sit on the ground for more than an hour, ready to be dispatched or fly on a mission, thus increasing the fuels temperature. Heat is the enemy of semiconductor electronics, resulting in errors or system failures.

The main benefit of moving to multicore processors is in reducing SWaP constraints. Multicore processors typically have better performance per watt than single core processors, and offer the most practical approach currently available to improve this feature. For example, the QorIQ processor will consume less than 30 watts of power with 8 cores running at 1.5 GHz, while the TILE64 processor consumes less than 22 watts of power with 64 cores each running at 700 MHz.

Multicore processors also offer faster communications across threads running on the different cores due to fast high-bandwidth internal networks and hardware cache-coherency protocols. Integration of I/O and memory controllers implies faster communications with external entities, while also reducing the need for additional board space for these components. Finally, consolidating the various processes on multicore processors results in fewer processors and boards. This helps to improve reliability and minimizes footprint which is reducing weight, board dimensions and susceptibility to different environmental conditions (e.g. vibration).

Multi-core FADEC can simplify the upgradeability of existing hardware with new functions, assuming some spare cores are left for planned upgrades. When accompanied with unambiguous definition of key system interfaces

among cores on a multicore chip and other distributed processing units, the plug-and-play philosophy in real-time deterministic environments can be supported.

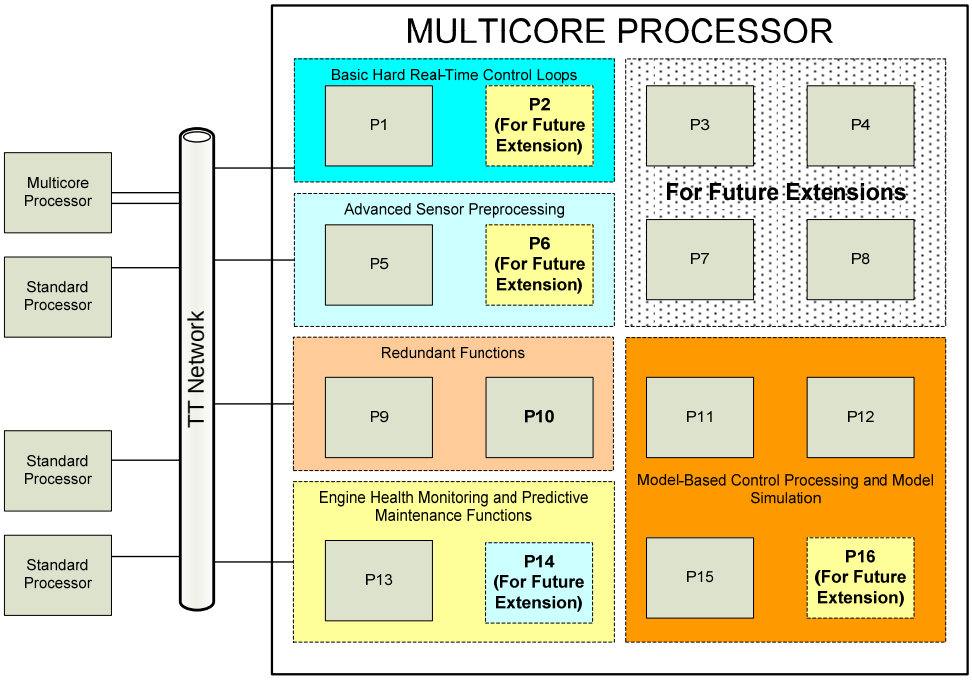

Figure 5. Hypothetical distributions of FADEC functions on a multicore in the hard real-time network environment

C. Multi-cores and processing for advanced controls

All of the above benefits can be combined to enable novel engine controller architectures. In the near term, processors with smaller numbers of cores can be used to extend existing controller architectures with new capabilities. Future cores with tens or hundreds of cores will necessitate re-thinking of the architecture to make effective use of all available cores. Possibilities include hosting individual threads on each core (data-flow architectures), thereby eliminating the need to schedule threads and improving time determinism.

One consideration when architecting future systems for multicore processors is the degree of replication. In order to maintain consistency among the replicas, this would be limited by the number of components (such as memory controllers) needed by all replicas. Another consideration is the level of replication – Replicas could be full replicas, duplicate controllers sharing a common model, or other combinations of the various components. Within each replica, the abundance of compute power enables the use of multiple simultaneous models, each fine-tuned for a different operating point, a different operating parameter, a different fault condition or some other factor. This vastly enhances the fidelity of the model, resulting in better control and operating efficiencies.

The generic example in Fig. 5 shows how the processing of complex functions can be split among processors on a multicore chip.

D. Software Considerations for Real-Time-System Design with Multicore-Processors

The apparent benefits of using multi-core processing must be tempered with current challenges in applying this technology in the aerospace environment. Since consumer-based computers usually run multiple applications in soft time, the benefits of multi-core processing can be more readily exploited. One important consideration when moving to multicore processors is that the software typically needs to be highly parallel to take full advantage of all the cores. Writing parallel code is difficult and error-prone. The software support for multicore processors has not kept pace with the developments in the hardware. While there is some emerging capability in parallel debugging and cycle-accurate simulation, these tools are inadequate. Multi-threading can introduce temporal interdependencies, can create races and deadlocks and also makes hard to verify and certify system software which runs in real or hard real-

time. Software for aerospace applications are intentionally designed to be more serial in execution and limited threads to create a more deterministic software code, which aids in the certification process.

The benefits of multi-cores can be fully exploited for aerospace applications only if every core provides deterministic processing "time budget" for each core with well-understood interaction points and data synchronization among tasks. This requires care to ensure synchronization to prevent processor stalls while awaiting the results of other threaded calculations, and greater expertise in error detection and debugging or multi-processors systems. So the first step in applying multi-core for aerospace applications is to design the software to take advantage of the multiple simultaneous processes, without having a significant negative impact on the certification process.

1. Operating System

Operating systems have not kept pace with developments in multicore processors to leverage their throughput capability. Current challenges include availability of certifiable operating systems (OS), which can properly take advantage of the multiple processors. Using partitioned software for FADEC applications is becoming more commonplace and will probably allow one possible way in which software elements can be targeted to processors. However, these types of a priori resource allocation often do not result in an optimal use of available processor resources.

Most of the operating systems currently available are not certifiable for multicore processors. Those that are certifiable for high-criticality applications only support Asymmetric Multi-Processing (AMP), but not Symmetric Multi-Processing (SMP). In AMP mode, a separate OS instance executes on each core and is responsible for scheduling threads on that core. In SMP mode, an OS instance is responsible for scheduling threads on multiple cores. The SMP mode is preferable as the OS can monitor all the cores and schedule threads on cores as they become idle. However, due to challenges such as schedulability, determinism and scalability, SMP operating systems are not yet usable in high-criticality systems. However there are other software design approaches which create controllable multiprocessors application execution[10], and execute well on heterogeneous systems with hierarchical memories (local and global shared memory) such as AMP (see Fig. 6).

A more efficient means to allocate resources would be at the thread level, but the availability of a certifiable OS that operates in such a mode is still to proven and the impact on the deterministic characteristics unclear, or in case of strict determinism probably impossible. Even with the availability of such an OS, the thread handling at the hardware level and the ability to prove determinism between multiple processors is also unclear and depends also on multi-core processor architecture.

2. Parallel Compilers

Tools and compilers can support design of parallel code, but as long as the programmer must manually define how an application is executed in parallel, the complexity and certification issues will delay use of multi-core processors in aerospace controls.

Compilers can simplify design of parallelized applications by using "sieves" [10] and standard serial source code. Sieves enable declarative concurrency which does not say how the code needs to be parallelized. It is only required to indicate which parts could be executed in parallel. This enables predictable execution and debugging of different architectures such as AMP er SMP without race conditions, typical for manually programmed threads or serial code without sieve construct. Sieve programs have same functional behvior on different architectures and can be debugged on a single core machine and deployed on e.g. an AMP. Execution time is different depending on the architecture for which the code is compiled, but due to deterministic declarative concurrency, mapping reliable execution on different processors is viable.

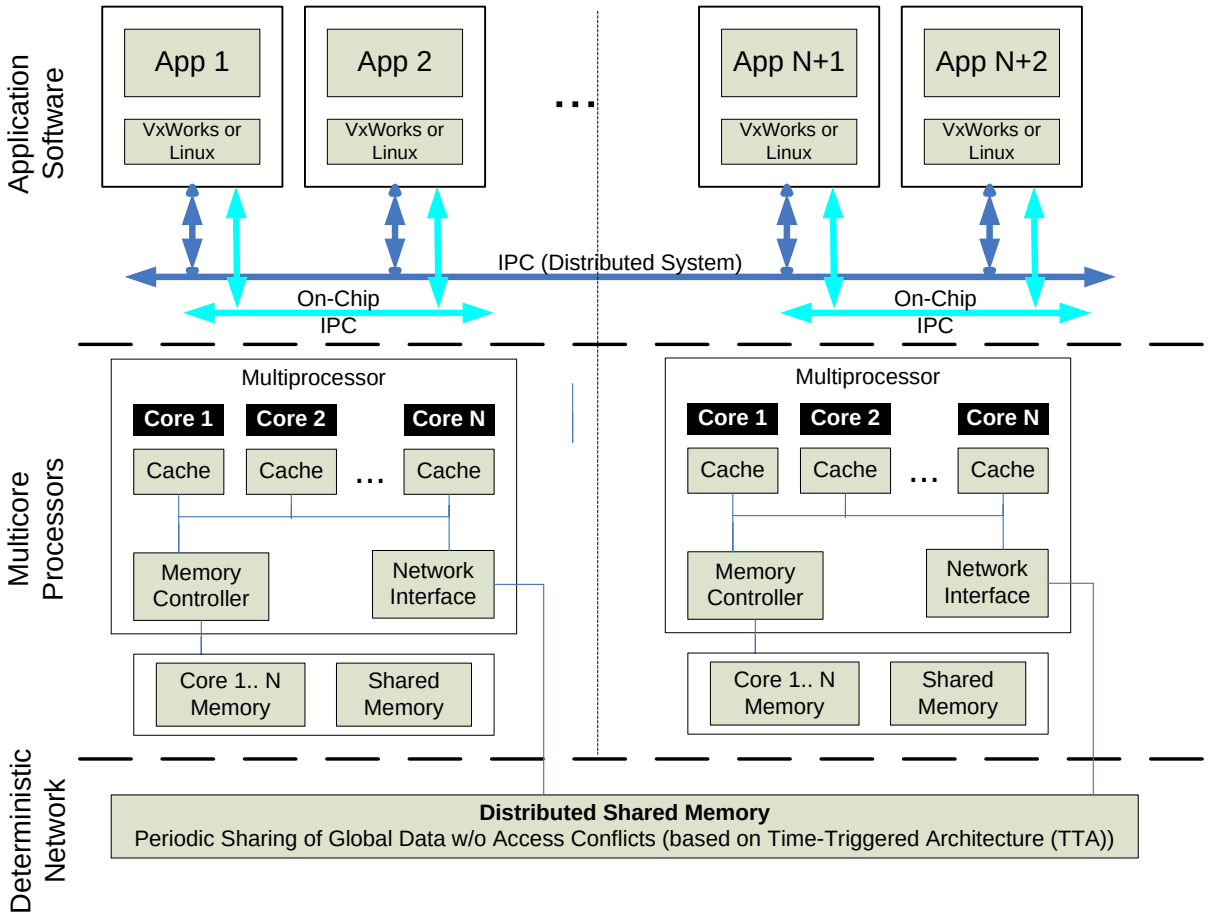

Figure 6. Asymmetric Multiprocessing (AMP) for distributed controls with multiprocessors: Memory and IPC model for distributed applications with multicore processor. Distributed modules are interconnected with time-triggered communication system (adapted and modified from Wind River whitepaper[3])

E. Designing deterministic distributed control system platforms with multi-core processors

Supporting multi-core MCUs, beyond two to four processing units will demand algorithms, languages, compilers and expertise in parallel computing that have not been previously necessary for aerospace applications. Broad experience in the area of hard real-time systems with predictable behavior is a not available, therefore some analogies can be a good starting point for advanced studies.

Many of the challenges with airworthy multi-tasking, multi computing, parallel processing in multi-core systems for hard real-time systems are addressed and solved in distributed system architecture such as Time-triggered Architectures (TTA)[10]. There are similarities between a set of processors operating on the shared memory, and a time-triggered architecture designed for fault-tolerant and distributed hard real-time systems (Figure 6) which operate using periodic message passing and local memories to share global variables.

This approach is similar to Asymmetric Multiprocessing (AMP) where[3]:

- All processors (or cores) can but they do not have to be identical (in TTA all distributed hosts can be different)
- Every processor has some private local memory and executes an independent instance of RTOS (as is in TTA)
- Inter-process communication (IPC) is provided, which allows data to be transferred and activities to be synchronized between processors (TTA have Time-triggered communication, scheduling of tasks in the system to the communications schedule)

- The system workload is defined at design time, with subsystems assigned to specific processors (in TTA all workload is scheduled at design time)
- The system is well-suited to applications where the need for data-sharing and interaction between subsystems is relatively limited (in a control system periodic message exchange limits communication upto Nx1000Hz)
- The system does not require applications to be multithreaded, since each application can run on a single processor subsystem; (no multi-threading in TTA, but a partitioned OSes can be used)
- Inter-Process Communication (IPC) is used to coordinate and combine the efforts of multiple processors (scheduled time-triggered communication is used for interprocessor and intertask communication)

In essence a time-triggered network can be seen as a heterogeneous multicore where the system architect can influence system architecture and design an AMP, where all distributed units exchange data over a shared system memory without access conflicts. Standard multicore processor hosts in a distributed system can be designed as a SMP or AMP system, depending on software architecture. The memory and data exchange architecture is presented in the figure 6.

The combination of partitioned OS such as VxWorks 653, hypervisor (controls different operating systems on different cores in a multi-core processor) to virtualize underlying computing multi-core platform, sieve-based compilers and time-triggered architecture for distributed system design could provide a basis for design of fully deterministic distributed FADEC systems with multicore processors. When used with model-based code generators, design and execution of complex operations can be simplified without non-deterministic race conditions among all tasks in the network.

V. Conclusion

In order to accommodate rising needs on operational efficiency and system optimization, FADECs will need to be integrated, distributed, and adaptive[12]. The performance pressures to improve FADEC processor throughput capability are mounting as next generation control algorithms become more demanding. Multi-processor capability is an enabling technology which is commercially available that can help add complex functions in the future FADEC designs.

Additional control system capabilities model-based simulations, sensor filter calculations, predictive maintenance and health management systems tend to consume more resources, and may add to the FADEC footprint and power dissipation if more powerful multicore processors are not used.

Multi-core processing offers attractive characteristics for improving overall throughput. The ability to take advantage of parallel processes without negatively impacting the certification of the system, is one of the key challenges that need to be worked out before the technology will be fully adopted for aerospace applications.

Multi-core processing used in deterministic distributed applications running on separate cores may require similar tools and design approaches known to designers of networked distributed real-time systems and TTA. This can make transition from centralized to distributed computing less complicated, at least from software perspective. Operating systems and parallel compilers will play the key role in future application of multi-core processors to advanced deterministic hard real-time controls such as FADEC. In using multi-cores and processing, the OS must be able to recognize multi-threading and the software must have simultaneous multi-threading technology (SMT) written into its code to enable parallel multi-threading wherein the cores are served multi-threaded instructions in parallel. Without SMT the software will only recognize one core. More work needed to take true advantages of multi-core processing for distributed engine FADEC applications.

References

Periodicals

Proceedings

[1]Kreiner A., Lietzau K., "The Use of On-Board Real-Time Models for Jet Engine Control", MTU Aero Engines, Germany, http://www.mtu.de/en/technologies/engineering_news/26869vki_kreiner_lietzau.pdf.

[2]Baptista M, Kumar A, Brunell B. and Viassolo D., "Model-Based Life Extending Control for Aircraft Engines", GE Global Research, Niskayuna, NY, AIAA-2004-6465 AIAA 1st Intelligent Systems Technical Conference, Chicago, Illinois, Sep. 20-22, 2004.

[3]Wind River, "WindRiver Solutions for Multiprocessing and Multicore" Whitepaper, Fig. 2, www.windriver.com.

[4]Mirko Jakovljevic M, Fulcher L., Behbahani A., Benson D., "Expectation and Vision for True modular Distributed Engine Control – Beyond 1st Project", 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, AIAA-2008-5279

[5] Link C. Jaw, Sanjay Garg, "Propulsion Control Technology Development in the United States - A Historical Perspective", NASA/TM—2005-213978, October 2005.

[6]Hall B., Paulitsch M., Benson D., Behbahani A, "Jet Engine Control Using Ethernet with a BRAIN,", 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, AIAA-2008-5291.

[7]Shaffer, P.L "Multiprocessor implementation of real-time control for a turbojetengine", IEEE Control Systems Magazine, Volume 10, Issue 4, Jun 1990 Page(s):38 – 42.

[8]Jakovljevic M., Artner, M., "Protocol-Level System Health Monitoring And Redundancy Management for Integrated Vehicle Health Management", Proceedings of IEEE Digital Avionics Systems Conference (DASC), Portland Oregon, Oct 2006.

[9]Kahle, J.A et al. „Introduction to the Cell Multiprocessor" IBM Journal of Research &Development, Vol. 49 No. 4/5 July/September 2005.

[10] Donaldson A., Lokhmotov A., Riley C., Cook A. "Auto-parallelisation of Sieve C++ Programs", Proceedings of the Euro-Par Workshop Highly Parallel Processing on a Chip (HPPC'07), Rennes, France, August 2007.

[11]TTA Project. The EU-funded OMI project 23396 TTA (Time-Triggered Architecture) aimed at the implementation of a time-triggered computer architecture (TTA) for fault-tolerant distributed real-time systems. http://www.vmars.tuwien.ac.at/projects/tta/index.html.

[12]Alireza Behbahani, Dennis Culley, Bert Smith, Christopher Darouse, Richard Millar, Bruce Wood, Jim Krodel, Sheldon Carpenter, Bill Mailander, Tim Mahoney, Ronald Quinn, Colin Bluish, Bobbie Hegwood, Gary Battestin, Walter Roney, William Rhoden, Bill Storey "Status, Vision, and Challenges of an Intelligent Distributed Engine Control Architecture", , 07ATC-267, SAE 2007 AeroTech Congress & Exhibition, September 17 – 20, 2007, Los Angeles, California, 2007-01-3859.

[13]Alireza Behbahani, "Adaptive Distributed Intelligent Control Architecture for Future Propulsion Systems", 61st Meeting Of The Society For Machinery Failure Prevention Technology, Virginia Beach, Virginia, Paper: Session 5B - Cape Colony B, April 17-19, 2007.

[14]Bhal Tulpule, Alireza Behbahani, and Richard Millar "Vision for Next Generation of Modular Adaptive Generic Integrated Controls (MAGIC) for Military/Commercial Turbine Engines", 43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 8 - 11 Jul 2007, Cincinnati Duke Energy Convention Center, Cincinnati, OH.

[15]Alireza R. Behbahani , "Achieving AFRL Universal FADEC Vision with Open Architecture Addressing Capability and Obsolescence for Military and Commercial Applications", , 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Sacramento, California, July 9-12, 2006.